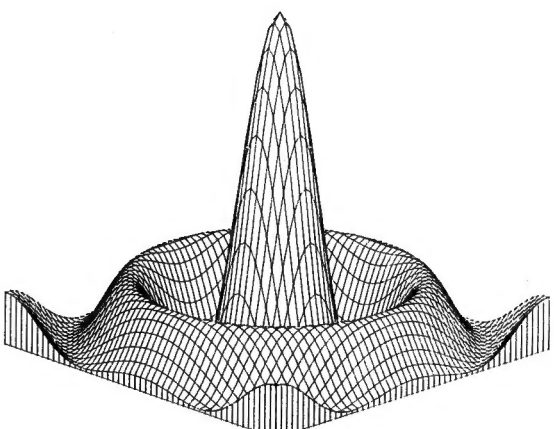
RESEARCH MACHINES

High Resolution Graphics



Reference Manual

High Resolution Graphics Reference Manual

Release 1, February 1980
(Revised November 1980)

The policy of Research Machines Limited is one of continuous development and improvement of its products and services, and the right is therefore reserved to revise this document or to make changes in the computer software it describes without notice. RML makes every endeavour to ensure the accuracy of the contents of this document but does not accept liability for any error or omission.

The original labelled distribution cassette tape or disc is regarded as the only proof of purchase and must be produced in order to qualify for an update at a reduced rate. Keep it safe and always work from copies.

Additional copies of this publication may be ordered from Research Machines at the address above. Please ask for "High Resolution Graphics Reference Manual".

HIGH RESOLUTION GRAPHICS
REFERENCE MANUAL

CONTENTS

CHAPTER 1

INTRODUCTION

This Manual accompanies the first release of the software package to support Research Machines' High Resolution Graphics. The routines to be described are supplied as extensions to BASIC or Algol 60, or as a graphics library for Fortran.

1.1 INTRODUCTION

High Resolution Graphics are implemented on the Research Machines 380Z by means of an additional board which is interfaced to the computer via the system bus. Although the board contains its own 16K bytes of memory, it shares address space with the 380Z VDU and the amount of address space available to the user for programs and data is not affected. The board can also operate when required as a 16K byte add-on memory (assuming the full complement of 56K bytes is not already available), but of course cannot be used for graphics at the same time.

In normal use the output from the High Resolution Graphics board is mixed with the output from the 380Z memory mapped VDU, allowing graphics and text to be superimposed, and the output from the board directly drives a black and white TV monitor. Small supplementary boards will soon be available to allow the use of a colour TV set or colour monitor.

The initial release of the software package supports the plotting of points, lines, and rectangular blocks at a number of intensities, these functions being available for the two resolutions at which the board operates. The brightness or colour of a set of picture elements of a given intensity can be changed instantaneously or gradually. In addition, the user may elect to subdivide the graphics memory into up to 8 logical views which can be displayed separately or in combination.

1.2 HOW TO USE THIS MANUAL

If your High Resolution Graphics board has been delivered separately from your computer, you will first have to install it. The instructions for doing this are in Appendix A.

You should next make a working copy of the graphics software package which will have been supplied as a special version of BASIC or Algol, or as a Fortran library.

Chapter 2 has been written primarily with the BASIC user in mind. It forms a tutorial introduction to the Graphics extensions to BASIC and we hope you will work through it at the computer, trying out all the examples. You should also study the example programs in Appendix B which demonstrate some of the effects that can be achieved.

Chapter 3 is the Reference Section of the manual. The various graphics procedures are described, ordered alphabetically. Two copies of a Quick Reference Guide are also provided in Appendix E. You may care to keep

one of these by the computer.

The remaining Appendices contain material that will be needed only occasionally. If you wish to try writing your own graphics support routines in Assembly language you will need to refer to Appendix C. Appendix D covers loading and saving pictures from disc. The tables in Appendix F are only needed for advanced graphics effects.

Algol and Fortran users are advised to read through Chapter 2 first, before concentrating on Chapter 3 which describes the available routines.

CHAPTER 2

GETTING STARTED WITH GRAPHICS

This Chapter is an introduction to the High Resolution Graphics support routines for the BASIC programmer. We suggest that you work through the Chapter at the computer. It is intended as a tutorial and you will probably only want to go through it once. After that the Reference Section in Chapter 3 and the Quick Reference Guide in Appendix E can be used as required.

Algol and Fortran users may also benefit from reading through this Chapter. If at all possible, they should try out the examples in BASIC, for the immediacy of an interpreted language encourages experimentation.

2.1 INITIAL TESTING

Enter the following BASIC program:

```
10 GRAPH 1: GRAPH 0
20 CALL "RESOLUTION",0,2
30 CALL "PLOT",0,0,3
40 CALL "LINE",150,90,3
```

and type RUN. A diagonal line should appear on the screen running from the left near the bottom to the middle. If you get an error message, type LIST and examine the program for typing errors. Pay special attention to the quotation marks surrounding the name of each CALLed routine and to the comma following.

If nothing appears to happen, work through the installation instructions again. In particular, check that the high resolution and VDU boards are correctly interconnected and (if using a black and white display) that you have moved the small video plug from the VDU board to the high resolution one.

If all is well, you are now ready to start programming with high resolution graphics. If you find the text going too fast, feel free to pause and try some experiments with what you have learned so far.

2.2 CLEARING THE SCREEN

Now have a look at the simple graphics program you have just typed in (type LIST). Notice that the program text and the high resolution display are superimposed. In line 10, the command GRAPH 1 clears the screen of text before plotting and confines subsequent text output to the bottom four lines of the screen, below the graphics area. GRAPH 0 allows text to occupy the full screen again. If you are unfamiliar with the effects of GRAPH 1 and GRAPH 0, try typing GRAPH 1 (then RETURN) followed by LIST. Then type GRAPH 0, then LIST again.

Another way to clear the screen of text is to type CTRL L (hold down the CTRL key, then type L) while BASIC is expecting a command (i.e. after

"Ready:"). On some keyboards, typing the key marked "PF" is equivalent.

During all of this the diagonal line has remained unchanged on the screen. This is because it is being produced from the high resolution board memory; this is entirely separate from the VDU memory from which text and low resolution graphics are displayed. One of the functions of the call to RESOLUTION in line 20 is to clear the high resolution memory. Try typing

```
CALL "RESOLUTION",0,2
```

(no line number). The line disappears. Note that you have just made a call to a graphics routine in "direct" mode (without a line number), as opposed to including the call in a stored program. This is often useful when experimenting.

Most users will find it quite convenient that the two displays share the same screen. In particular, text can be output by the PRINT and PLOT statements (to the standard 380Z low resolution screen) to annotate a high resolution graph.

2.3 SETTING THE RESOLUTION

Another function of RESOLUTION is (as suggested by its name) to set the resolution of the graphics display. Two modes are available, "high resolution" mode (HR), in which the display resolution is 319 horizontally by 192 vertically, and "medium resolution" (MR) with a resolution of 160 horizontally by 96 vertically. A call to RESOLUTION is of the general form:

```
CALL "RESOLUTION",R,B
```

where R is a number or numeric expression that can take the value 0 or 1 and determines whether the resolution will be high or medium, respectively, and B sets the size of a picture element (see below). Thus the call to RESOLUTION in the example program had "argument" R equal to zero and therefore set high resolution mode. Try changing line 20 to

```
20 CALL "RESOLUTION",1,4
```

and then RUNning the program again. As can be seen, the diagonal line is twice as long as we are now in medium resolution mode; it is also rather dim as its intensity is only 3 compared to a maximum of 15.

The second argument of the call to RESOLUTION sets the number of bits to be used for each picture element or "pixel". A pixel is that element of graphics memory that determines the intensity of a single point. In HR mode, two bits are available, giving a choice of 4 intensities (colours or shades of grey) since the two (binary) bits can take on values of 0, 1, 2 or 3. With MR, four bits are available allowing you to choose from 16 intensities. The effect of restricting the pixel to less than two or four bits will be described later.

A call to RESOLUTION should always be made before starting a picture and should be thought of as initialising the graphics system. Both arguments

must be supplied, otherwise the error message

```
Bad no of args
```

will result.

2.4 PLOTTING POINTS AND LINES

Lines 30 and 40 of the example program are responsible for drawing the diagonal line. PLOT causes a dot to be plotted; its general form is

```
CALL "PLOT",X,Y,I
```

which plots a dot of intensity I at position (X,Y). The origin, position (0,0), is set by RESOLUTION to the bottom left corner of the graphics area. In HR mode, the other three corners are, going clockwise, (0,191), (318,191) and (318,0). Change line 20 of the program back to:

```
20 CALL "RESOLUTION",0,2
```

then change line 40 onwards to:

```
40 CALL "PLOT",0,191,1
50 CALL "PLOT",318,191,2
60 CALL "PLOT",318,0,3
```

then RUN. The program puts a dot in each corner of the screen. The bottom left dot is bright (intensity 3), as is the bottom right one. The top left one should be rather dim (intensity 1) and the top right one somewhat brighter. If you cannot see all four dots, try adjusting the brightness and contrast controls of your TV monitor.

Like PLOT, the call to LINE is of the general form:

```
CALL "LINE",X,Y,I
```

but now (X,Y) specifies the end point of a line. The start point is given either by a call to PLOT, or by the end point of a call to LINE, whichever occurred most recently. (RESOLUTION initialises these "memorised" XY coordinates to (0,0).) To draw a rectangular box enclosing the screen, change PLOT to LINE in lines 40 to 60:

```
40 CALL "LINE",0,191,1
50 CALL "LINE",318,191,2
60 CALL "LINE",318,0,3
```

and add

```
70 CALL "LINE",0,0
```

and RUN. (Now might be a good moment to investigate BASIC's EDIT command if you are not already familiar with it!)

Note that the omission of the intensity argument in line 70 is deliberate. It can be left out in calls to both PLOT and LINE; the most

recently supplied value is used, in this case intensity 3. RESOLUTION sets the "memorised" intensity to zero.

There are two important reservations which should be borne in mind when using the call to LINE. Firstly, the LINE call does NOT plot the first point of the line, which is assumed to have been output by a preceding call to PLOT or LINE. If LINE replotted it, exclusive OR plotting of lines (described below) would not work correctly. Secondly, in general, a line drawn in one direction will NOT be identical to a line joining the same two points but drawn in the opposite direction. This arises as a result of the high speed line drawing algorithm used. If you later wish to erase a line, make sure you draw over it in the same direction as it was originally drawn.

2.5 RANGE OF XY COORDINATES

As already mentioned, the visible screen extends in HR from (0,0) to (318,191). The values of X and Y passed in calls to PLOT and LINE are not restricted to this range, however, and in fact X and Y can specify any point on a "virtual" screen which extends from -32768 to +32767 in both axes. You can demonstrate this by adding:

```
80 CALL "PLOT",1000,1000,3
90 CALL "LINE",-1000,-1000
```

then typing RUN. A line will be drawn through the origin at intensity 3.

If the X or Y value (or indeed any value in a CALL) is greater than 65535 the error message

```
Illegal function
```

is displayed, whilst if it is less than -65535, the message

```
Syntax error
```

appears. Values between +32767 and +65535 are treated by PLOT and LINE as though 65536 had been subtracted first, and similarly, values from -65535 to -32769 have 65536 added. To avoid confusion you are advised only to use values in the range -32767 to +32767.

2.6 MOVING THE ORIGIN

In order to simplify graphics programs the OFFSET call allows the visible screen to be moved with respect to the virtual screen by setting the XY coordinates of the lower left corner of the visible screen. Its general form is:

```
CALL "OFFSET",X,Y
```

If for example the statement:

```
25 CALL "OFFSET",-50,-50
```


is added to the program, the lines that met at (0,0) move 50 screen units upwards and to the right. Try this first, then try

```
25 CALL "OFFSET",-1050,-1050
```

You can now see the end point of the long line drawn by line 90 although none of the rectangle is visible.

Note that OFFSET does not move any graphics already displayed; it merely resets the visible screen for graphics drawn subsequently. You can verify this by deleting line 25 and typing the same statement at line 75. Both parts of the picture are now seen. Note also that RESOLUTION always sets or resets the offset to (0,0).

OFFSET is sometimes convenient when plotting graphs. Try the following program:

```
10 GRAPH 1: GRAPH 0
20 CALL "RESOLUTION",0,2
30 CALL "OFFSET",0,-96
40 LET A=16*ATN(1)/318
50 FOR X=0 TO 318
60 LET Y=50*SIN(A*X)
70 CALL "PLOT",X,Y,3
80 NEXT X
```

which plots a sine wave across the middle of the screen. In this case OFFSET avoided the need to add a value to the Y coordinate calculated at line 60.

If you are ready for a rest, you might like now to have a look at the STAR program (listed in Appendix B) which illustrates the use of the calls so far described.

2.7 BLOCK FILL

The FILL call provides a means for the rapid filling of a rectangular area of the screen, for example for drawing histograms. Its general form is

```
CALL "FILL",X1,Y1,X2,Y2,I
```

This fills in the area whose lower left and top right corners are specified by the points (X1,Y1) and (X2,Y2), respectively. To draw a filled rectangle X2 should be greater than X1 and Y2 should be greater than Y1. If either X1,X2 or Y1,Y2 are the same then a vertical or horizontal line is drawn. If both X1,X2 and Y1,Y2 are the same then a dot is drawn. If X2 is less than X1 or if Y2 is less than Y1 then nothing is drawn.

As with PLOT and LINE, the intensity argument I can be omitted; if so the most recently used value applies

See Appendix B (COSINE HISTOGRAM) for an example of block fill.

2.8 THE INTENSITY ARGUMENT

In the examples so far, the intensity argument I has ranged from 0 to 3 in high resolution or from 0 to 15 in medium resolution, where 0 corresponds to the background (initially black) and the non-zero values to progressively lighter shades of grey. In this mode plotted data overwrites any preexisting information. For example:

```
10 CALL "RESOLUTION",0,2
20 CALL "FILL",50,50,150,150,2
30 CALL "PLOT",0,0,3
40 CALL "LINE",200,200
50 INPUT A
60 CALL "PLOT",0,0,0
70 CALL "LINE",200,200
```

This program first draws a grey square then overwrites it with a diagonal white line. Line 50 merely causes a pause until a numeric value is entered. The white line is erased by replotting it at intensity zero, leaving however a "black" line through the grey square.

This can be avoided by using EXCLUSIVE OR plotting. Considering two binary digits (bits) A and B, the truth table of the XOR function is:

A	B	A XOR B
0	0	0
1	0	1
0	1	1
1	1	0

If we consider A to be the intensity bit we are plotting and B to be the existing screen content, it can be seen that when plotting a one, if the corresponding bit of the screen is off, it is turned on (set to one), while if it is already on it is turned off (set to zero). The important feature of XOR plotting occurs when a point is replotted in XOR mode at the same intensity. This is equivalent to combining A with A XOR B in the above table. It can be seen that the new result is identical with B. In other words, a point plotted in XOR mode is reversible without loss of the original information.

XOR plotting takes place when the intensity argument is negative. In high resolution values for I of -1 to -3 correspond to values of +1 to +3 but are plotted by exclusive OR rather than by replacement.

To demonstrate this, change lines 30 and 60 to:

```
30 CALL "PLOT",0,0,-3
60 CALL "PLOT",0,0,-3
```

Now, when the original line is replotted, the square is left intact.

XOR plotting is useful for drawing data provisionally (e.g. an "elastic band" line from a fixed point to a roving cursor) or for temporarily highlighting an area by means of an XOR block fill. Note that both intensity settings have the same (negative) value for I, as opposed to

replacement plotting when the two (positive) values are different.

The example program CUBISM in Appendix B uses XOR plotting.

2.9 PEN UP MOVEMENTS

It is occasionally convenient to reset the memorised (X,Y) coordinates (the start point of a line) without causing any graphical output. This occurs when an intensity argument of 16 is used.

2.10 MODIFYING THE DISPLAYED INTENSITY

A powerful feature of the Research Machines Graphics Board is the ability to change the displayed brightness that corresponds to one of the plotted intensities virtually instantaneously. For example, intensity 3 could first be set to display at the same level as the background (e.g. black). A complex picture could then be drawn, perhaps taking several seconds. Finally intensity 3 could be set to display as white, causing the complete picture to appear as a whole in an instant.

The mapping of the plotted intensity (in the range 0 to 3 or 0 to 15 as previously described) to the displayed intensity, a value from 0 to 255, corresponding to shades of grey from black to white, is carried out by means of a special hardware area on the graphics board called the colour lookup table and the values this contains are set by the COLOUR call.

A call to RESOLUTION sets up default values in the colour lookup table. For example, in high resolution, the values assigned to plotted intensities of 0, 1, 2 and 3 are 0, 64, 128 and 255, corresponding to black, dark grey, light grey and white. In medium resolution, the intensity arguments 0 to 15 are similarly mapped to black and 15 progressively lighter shades of grey, again arranged in an approximately logarithmic series, which the eye sees as roughly equal steps of brightness.

These default values can be changed at any time by a call to COLOUR. The call is of the general form

```
CALL "COLOUR",I,N
```

where I is the plotted intensity value (range 0 to 3 for high resolution, 0 to 15 for medium) and N is the new displayed intensity (range 0 to 255). For example, in high resolution, the statements

```
100 CALL "COLOUR",0,255
110 CALL "COLOUR",1,128
120 CALL "COLOUR",2,64
130 CALL "COLOUR",3,0
```

would reverse any graphics currently displayed, causing the appearance of a "negative". Note that a call to COLOUR can easily be placed in a loop, allowing a picture to be faded up or down. For example

```
140 FOR N=1 TO 255
```



```
150 CALL "COLOUR",3,N
160 NEXT N
```

fades intensity 3 up from 0 (black) to 255 (white) again.

A call to COLOUR with no arguments, viz.

```
170 CALL "COLOUR"
```

restores the default values as set by RESOLUTION.

2.11 SETTING THE COLOUR

In the description so far, it has been assumed that the graphics output is being displayed in black and white. The graphics board can also display in colour, provided a colour modulator and a colour TV set or monitor are available. As described in the previous section, the hardware colour look up table maps the plotted intensity (pixel information) to a value between 0 and 255, that is, to an 8 bit value. When a colour modulator is installed, 3 of these 8 bits are assigned to red, 3 to green and 2 to blue, giving ranges for red, green and blue of 0 to 7, 0 to 7 and 0 to 3, respectively. To make the setting up of a given colour easier, the COLOUR call has an alternative form

```
CALL "COLOUR",I,R,G,B
```

where I corresponds to the plotted intensity as before and R, G and B are the required values for red, green and blue. Just as for black and white, this allows the colour associated with a given plotted intensity to be changed.

2.12 PROGRAMMING MORE THAN ONE PICTURE

So far the discussion has been confined to a single picture. In high resolution with two bits/pixel only one picture is available; the whole of the usable graphics memory is needed. In the other mode mentioned, medium resolution with four bits/pixel, only half of the graphics memory is used and two separate pictures can be constructed.

These are considered to occupy two "pages" of the (medium resolution) graphics memory and the pages are numbered 0 and 1. It is important to note that you can display either page 0 or page 1, but never both together. (This is a consequence of the way the graphics board is designed.) After a call to RESOLUTION, page 0 is selected both for display and for update. Calls to PLOT, LINE and FILL write to page 0 and the display circuits show the contents of page 0 on the screen.

The page on which plotting takes place can be changed by a call to UPDATE of the general form

```
CALL "UPDATE",P
```

where P has the value 0 or 1, and similarly, the page that is displayed can be changed by a call to DISPLAY of the form


```
CALL "DISPLAY",P
```

Notice that the separation of the update and display functions allows you to display one picture while generating another, thus allowing a degree of animation without the relatively slow process of picture generation being visible.

The following simple example clarifies this:

```
10 GRAPH 1: GRAPH 0
20 CALL "RESOLUTION",1,4
30 CALL "UPDATE",0
40 CALL "PLOT",0,0,8
50 CALL "LINE",159,95
60 CALL "UPDATE",1
70 CALL "PLOT",0,95,15
80 CALL "LINE",159,0
90 INPUT "Display which page";P
100 CALL "DISPLAY",P
110 GOTO 90
```

Page 0 displays a medium grey line from bottom left to top right, page 1 a white line from top left to bottom right. Line 90 allows you to select one or the other. Type CTRL Z to exit.

You are reminded that these two pages are only available in medium resolution.

2.13 USING A REDUCED NUMBER OF BITS/PIXEL

The graphics software also supports three additional modes in which the number of bits/pixel is reduced. The effect of this is to restrict the number of separate intensities that can be displayed and instead to allow multiple pictures to be generated. In a way the result is similar to the concept of "pages" in medium resolution which has just been introduced but since the effect is available in high resolution as well and since the mechanism is different, the name "view" has been chosen. Considering high resolution, we can either select 2 bits/pixel (as we have done so far) or 1 bit/pixel and have two views. Like pages, views are normally displayed separately, but since view selection is carried out by software manipulation of the colour lookup table, it is possible to arrange to display more than one view at once.

As indicated earlier, the choice of number of bits/pixel is made by the arguments used in a call to RESOLUTION. Thus the call

```
10 CALL "RESOLUTION",0,1
```

sets high resolution with one bit/pixel and two views, view 0 and view 1. The choice of view is made in the same way as for page in medium resolution; thus CALL "UPDATE",1 selects view 1 for PLOT, LINE and FILL and CALL "DISPLAY",1 results in view 1 being displayed. As before, RESOLUTION automatically selects view 0 both for update and display. Note that in this mode, the intensity argument I can only take on values of 0 or 1. (In fact the I argument is ANDed with the maximum value

allowed, so that if, for example, 2 is supplied, 2 AND 1 (= 0) is used for plotting.) However as before the shades of grey or colours that correspond to the two intensities can be changed by calls to COLOUR. For example

```
10 CALL "RESOLUTION",0,1
20 CALL "COLOUR",0,0,0,3
30 CALL "COLOUR",1,7,0,0
40 CALL "PLOT",0,0,1
50 CALL "LINE",318,191
```

will plot a red diagonal line on a blue background as view 0.

2.14 MULTIPLE VIEWS IN MEDIUM RESOLUTION

The remaining two modes that can be selected are both in medium resolution. If the pixel is restricted to one bit by a call of the form

```
10 CALL "RESOLUTION",1,1
```

we still have two pages but each contains four views, making eight pictures in all. The intensity argument may take on values of 0 or 1. Calls to UPDATE and DISPLAY require two arguments and are of the form:

```
CALL "UPDATE",P,V
CALL "DISPLAY",P,V
```

where P is the page (range 0 to 1) and V is the view (range 0 to 3). The N argument in a call to COLOUR must lie in the range 0 to 1.

Similarly, a two bit pixel can be selected, giving a choice of two pages, each with two views and four intensities.

The five possible modes are conveniently referred to as HR2 (high resolution, 2 bits/pixel), HR1, MR4, MR2 and MR1.

Notice that although UPDATE and DISPLAY accept a variable number of arguments (page, view or both, depending on mode) both page and view can always be supplied without confusion. In MR4 for example, the view argument is ignored if supplied, while in HR1 the page argument is discarded.

The program REVOLVE in Appendix B shows how eight medium resolution pictures can create the illusion of motion when displayed in rapid succession.

2.15 CLEARING A VIEW

When working with more than one view it is frequently desirable to clear a view prior to updating it. (This will usually be done invisibly in the "background" by selecting another completed view for display.) One means of doing this, having selected the view for update, is simply to block fill it with intensity 0. For example, in HR1, the statements


```
50 CALL "UPDATE",1
60 CALL "FILL",0,0,318,191,0
```

would clear view 1. As a convenience, the call to CLEAR has the same effect. Thus the above example can be replaced by

```
50 CALL "UPDATE",1
60 CALL "CLEAR"
```

CLEAR absolves the programmer from having to be aware (at that moment) of the current offset or resolution.

2.16 DISPLAYING MORE THAN ONE VIEW

This Section can be skipped on first reading; it is only necessary to study it when you wish to display more than one view at once.

Multiple views (but NOT multiple pages) are achieved by manipulation of the colour lookup table. Consider high resolution; normally, with 2 bits/pixel, intensities 0 to 3 are arranged to map to black, dark grey, light grey and white in the colour lookup table (CLT). We may represent this as:

(a) PIXEL	CLT
00	0
01	64
10	128
11	255

If instead we arrange that only pixels in which the less significant bit are set are displayed at an intensity different from the background, these become the only pixels that are visible. The colour lookup table might become:

(b) PIXEL	CLT
00	0
01	200
10	0
11	200

Information represented by the more significant bit is not differentiated from background (except where the less significant bit is also set). Alternatively we can arrange to suppress the information carried in the less significant bit:

(c) PIXEL	CLT
00	0
01	0
10	200
11	200

and only see the more significant bit.

This is the means used to achieve the separate views. After a call to RESOLUTION which specifies a restricted number of bits/pixel (modes HR1, MR1 and MR2), DISPLAY swaps around the colour lookup table in the manner outlined above, UPDATE arranges that PLOT, LINE, FILL and CLEAR only modify the appropriate part of each pixel and COLOUR modifies the entries in the colour lookup table appropriate to the currently selected view. Thus in the illustrations of colour table organisation above, intensity 0 corresponds to a displayed intensity of zero and intensity 1 to 200, but the positions in which 0 and 200 appear depend on whether view 0 or view 1 is selected for display.

It is by manipulation of these positions that the programmer can select a combination of views. Thus to display both view 0 and view 1, the colour table is changed to:

(d) PIXEL	CLT
00	0
01	200
10	200
11	200

while to display only those pixels which are common to view 0 and to view 1, the table is:

(e) PIXEL	CLT
00	0
01	0
10	0
11	200

It should be realised that case (a) is automatically selected when the resolution is 2 bits/pixel, and that with 1 bit/pixel, cases (b) and (c) occur automatically as a result of a call to DISPLAY requesting view 0 or 1. It is only when case (d) or case (e) is required that programmer intervention is necessary.

Two additional calls allow direct manipulation of the colour table. SETCOL sets a specified element of the colour table to a display intensity value. Its general form is the same as that for COLOUR, namely:

```
CALL "SETCOL",I,N
```

where I is the "address" in the colour table to modify and N is the new value. Like COLOUR, the alternate form

```
CALL "SETCOL",I,R,G,B
```

is available for use with a colour monitor.

Calls to SETCOL do not actually effect any change in the contents of the colour table, they merely set up the values in memory (in a special copy of the colour table which is maintained in all modes). When all the values to be changed have been set up, the colour table is loaded by a

call to VIEW of the form:

```
CALL "VIEW",P
```

The P argument indicates which medium resolution page is to be displayed. It can be left out in high resolution but must be supplied in medium.

When using SETCOL and VIEW it is important for the programmer to realise that he is taking over functions that are normally carried out automatically. Thus in HR modes he must set up four intensity entries with SETCOL even when, as in mode HR1, there are only two logical intensities in use. Similarly, in the MR modes, all sixteen intensity entries must be defined. (The logical intensities are normally multiplexed into the colour lookup table by DISPLAY or COLOUR.)

The following program selects mode HR1, setting up case (d) with view 0 and view 1 displayed simultaneously:

```
10 CALL "RESOLUTION",0,1
20 DATA 0,200,200,200
30 FOR I=0 TO 3
40 READ N
50 CALL "SETCOL",I,N
60 NEXT I
70 CALL "VIEW"
```

You need only change line 20 to

```
20 DATA 0,0,0,200
```

to set up case (e), displaying instead those pixels that are common to view 0 and view 1.

Note that this method is also of general application in all modes. It provides a means of deferring the alteration of the colour table until all the colours have been selected. This is not normally of concern in high resolution, but in medium resolution, the changing of all 16 colours by use of the COLOUR call does cause a discernible "ripple" owing to the finite time a call to COLOUR takes. (This is partially because COLOUR waits for the end of a frame, up to 20 ms, before changing the colour table. Sixteen calls to COLOUR therefore take at least 320 ms.) Instead, then, the sixteen colours can be selected by SETCOL then finally loaded into the table by VIEW, thus changing all the displayed colours at once. Examples of both methods follow:

```
10 CALL "RESOLUTION",1,4
20 DATA ...specifying the colours
30 FOR I=0 TO 15
40 READ R,G,B
50 CALL "COLOUR",I,R,G,B
60 NEXT I
```

```
10 CALL "RESOLUTION",1,4
20 DATA ...specifying the colours
30 FOR I=0 TO 15
```



```
40 READ R,G,B
50 CALL "SETCOL",I,R,G,B
60 NEXT I
70 CALL "VIEW",0
```

Remember that VIEW must be followed by page in medium resolution.

User alteration of the colour table is more complicated in modes MR1 and MR2 than in mode HR1 but follows the same general principles. Appendix F contains tables showing which colour table elements affect which view.

The example program GRAPH in Appendix B illustrates the use of SETCOL and VIEW in displaying two views simultaneously.

2.17 SUMMARY

The following is a summary of the concepts presented in this Chapter:

The high/medium resolution and low resolution displays are separate. Low resolution text (or graphics) can be superimposed on a high or medium resolution picture.

The low resolution display is cleared by the GRAPH 1 statement or by PRINTing the form feed character (CHR$(12)) (GRAPH 0 mode only). The high resolution display is cleared by calling RESOLUTION or CLEAR (the latter clearing only the current page and view).

The high resolution board can be set to high resolution (319 by 192) or to medium (160 by 96) by calling RESOLUTION with the appropriate arguments.

The XY coordinates of the lower left corner of the screen are initially (0,0) but can be changed by calling OFFSET.

A point is plotted by calling PLOT with arguments X, Y and I. For the point to be visible its XY coordinates must lie within the screen limits defined by RESOLUTION and OFFSET. The logical intensity I must lie in the range 0 to 3 (high resolution) or 0 to 15 (medium resolution).

A line is plotted by calling LINE with arguments as for PLOT. Its start point is the most recently plotted point or the end point of the most recent line; its end point is passed in the call. Intensity as for PLOT.

A rectangular block can be filled by calling FILL. Two pairs of XY coordinates specify the lower left and upper right corners of the filled area. Intensity as for PLOT.

The actual (physical) intensity of a point, line or block that corresponds to its logical intensity can be altered to any value in the range 0 to 255 by calling COLOUR. An alternate form of the call allows red, green and blue intensities to be set individually in colour systems.

Normal plotting is by replacement. If the intensity is negative plotting is by exclusive OR, allowing graphics to be "unplotted" without loss of the original screen contents.

If the intensity is 16 the XY coordinates are set but there is no output.

In high resolution the number of bits/pixel can be reduced from 2 to 1, giving 2 views but reducing the number of intensities from 4 to 2. Calling DISPLAY selects the view to display. PLOT, LINE and FILL affect the view selected by UPDATE.

In medium resolution two separate pages are available, selectable by DISPLAY and UPDATE. Reducing the number of bits/pixel to 2 or 1 gives 2 or 4 views/page, with 4 or 2 intensities.

More than one view can be displayed by modifying the colour lookup table with SETCOL. The changes become visible after calling VIEW.

Saving and loading pictures from disk is described in Appendix D.

2.18 CONCLUSION

Now that you have worked through all the graphics calls you should be ready to start experimenting with your own graphics programs. A first step might be to study the examples given in Appendix B. These are not meant to be particularly serious but they do give an idea of some of the effects that can be achieved.

PAGE INTENTIONALLY BLANK

CHAPTER 3

REFERENCE SECTION

This Chapter contains the formal definitions of the graphics routines that can be called from BASIC, Algol and Fortran. The descriptions are in alphabetical order and are set out as follows:

Form: shows the correct form for calling the routine. Where the routine is available for more than one language the form for each language is shown.

Purpose: a brief summary of what the routine is used for.

Remarks: describe in detail how to use the routine.

Examples: contain sample program fragments that demonstrate how to call the routines.

Some special terms are used in the definitions. Here is a brief glossary:

Pixel - a picture element. The number of pixels in a picture depends on the resolution selected. The size of each pixel (in bits) governs the number of intensities it can take on.

Resolution - the HRG board can operate in high resolution (319 by 191 pixels per picture) or medium resolution (160 by 96).

Intensity - the intensity range of each pixel depends on its size in bits. For example, with 2 bits/pixel, its intensity can be 0, 1, 2 or 3.

Colour - a term used to describe the physical brightness that corresponds to a given intensity. It can range from 0 to 255 (black to white). In colour systems red, green and blue components can be specified.

Colour lookup table - a part of the HRG hardware in which each intensity is mapped to its corresponding colour.

Page - two separate pages of graphics memory (two separate pictures) are available in medium resolution.

View - by restricting the number of bits/pixel it is possible to have more than one picture on each page. These "logical" pages are called views.

3.1 CLEAR (BASIC only)

Form:
BASIC: CALL "CLEAR"

Purpose: To clear the currently selected page and view.

Remarks: CLEAR clears the page and view selected by UPDATE to logical intensity 0. It works by making an internal call to FILL and absolves the programmer from having to be aware of the current resolution and offset. It is useful when clearing selected views but calling RESOLUTION is faster when it doesn't matter that the whole of graphics memory is cleared.

Example:
BASIC:
```
10 CALL "RESOLUTION",1,1
20 CALL "UPDATE",1,3
30 CALL "CLEAR"
```

clears view 3 of page 1.

3.2 COLOUR (BASIC)

Forms:
BASIC:
```
CALL "COLOUR",I,N
CALL "COLOUR",I,R,G,B
CALL "COLOUR"
```

Purpose: To set the actual intensity or colour displayed on the screen that corresponds to a logical intensity I.

Remarks: COLOUR sets the value in the colour lookup table that corresponds to the logical intensity I plotted by PLOT, LINE or FILL.

The first form with two arguments is used with black and white displays. The value of N can range from 0, which displays as black, to 255, displaying as white, with a continuous range of progressively lighter shades of grey in between.

The second form with four arguments is for colour displays. R, G and B set the red, green and blue intensities. The red and green intensities can range from 0 to 7, the blue from 0 to 3.

Calling RESOLUTION sets up default values in the colour lookup table such that increasing logical intensity corresponds to approximately equal steps of increasing brightness. (These default values are in general only suitable for black and white displays.)

The third form of the call to COLOUR with no arguments restores the default values set by RESOLUTION.

Example:
BASIC:
```
10 CALL "RESOLUTION",0,2
20 DATA 255,128,64,0
30 FOR I=0 TO 3
40 READ N
50 CALL "COLOUR",I,N
60 NEXT I
```

sets the colour lookup table to produce a reversed image. (The default values for the lookup table in mode HR2 are 0, 64, 128 and 255.)

3.3 COLOUR (Algol, Fortran)

Form:
Algol: colour(location(a[0]))

Fortran: CALL COLOUR(A)

Purpose: To load the colour lookup table from byte array a or A.

Remarks: The COLOUR call sets up values in the colour lookup table that determine the actual intensities or colours that correspond to the logical intensities plotted by PLOT, LINE and FILL.

The address of a byte array a (or A) is passed to the COLOUR procedure. It is the programmer's responsibility to set up the contents of this array before calling COLOUR. The array is normally of dimension 16 bytes to suit all resolutions. COLOUR copies as many elements as necessary from the array to the colour lookup table. The number of elements copied depends on resolution and bits/pixel, being 2 in modes HR1 and MR1, 4 in HR2 and MR2 and 16 in MR4.

The value stored in each element can range from 0 (black) to 255 (white). For colour use the procedure (or function) MIX to combine the desired red, green and blue components into a single value.

Calling RESOLUTION sets up default values in the colour lookup table, such that increasing logical intensity corresponds to approximately equal steps of increasing brightness. COLOUR allows these to be changed. It should be compared with VIEW which performs a similar function but loads the colour lookup table directly.

Examples:
Algol:
```
BYTE ARRAY a[0:15];
resolution(0,2);
a[0] = 255; a[1] = 128;
a[2] = 64; a[3] = 0;
colour(location(a[0]));
```

Fortran:
```
BYTE A(16)
DATA A/255,128,64,0,12*0/
CALL RESOL(0,2)
CALL COLOUR(A)
```

set the colour lookup table to produce a reversed image. (The default values for the lookup table in mode HR2 are 0, 64, 128 and 255.)

3.4 DISPLAY

Forms:

```
BASIC:    CALL "DISPLAY",P,V     (all modes)
          CALL "DISPLAY",V       (HR 1)
          CALL "DISPLAY",P       (MR 4)

Algol:    display(p,v)

Fortran:  CALL DISPLY(IP,IV)
```

Purpose: To display page P, view V.

Remarks: In mode HR2 there is only one page and view and this call is not used. In all MR modes there are two separate pages, numbered 0 and 1. In modes HR1 and MR2 there are two views, numbered 0 and 1, and in mode MR1 there are four views, numbered 0 to 3.

After a call to RESOLUTION, page 0 view 0 is selected for display. Calling DISPLAY changes the page and/or view of graphics memory that is displayed. Similarly, calling UPDATE selects the page and view that will be modified by PLOT, LINE and FILL.

In BASIC only, P may be omitted in mode HR1 and V in mode MR4.

Examples:

```
BASIC:    10 CALL "RESOLUTION,1,1
          20 CALL "DISPLAY",1,3

Algol:    resolution(1,1);
          display(1,3);

Fortran:  CALL RESOL(1,1)
          CALL DISPLY(1,3)
```

set mode MR1, then select page 1, view 3 for display.

3.5 FILL

Forms:

BASIC: CALL "FILL",X1,Y1,X2,Y2,I
CALL "FILL",X1,Y1,X2,Y2

Algol: fill(x1,y1,x2,y2,i)

Fortran: CALL FILL(IX1,IY1,IX2,IY2,I)

Purpose: To fill the rectangle whose lower left corner is (X1,Y1) and upper right corner is (X2,Y2) with intensity I.

Remarks: The point (X1,Y1) specifies the lower left corner and the point (X2,Y2) the upper right corner of a rectangle to be filled in. These XY values can lie in the range -32768 to +32767 but extend only to the edge of the screen. Thus X1 can never be less than the left hand border, X2 never more than the right border, Y1 never less than the lower border and Y2 never more than the upper border. If X1 and X2 are the same but Y2 is greater than Y1, a vertical line is drawn, and if Y1 and Y2 are the same but X2 is greater than X1, a horizontal line is drawn. If X1 is the same as X2 and Y1 is the same as Y2, a point is plotted. If X2 is less than X1 or if Y2 is less than Y1 then there is no output.

I determines the logical intensity of the block in the same way as for PLOT (see Section 3.11). If omitted (BASIC only), the value last specified in a call to PLOT, LINE or FILL is used.

The "memorised" XY coordinates are not affected by FILL.

Examples:

BASIC: 10 CALL "FILL",50,50,100,100,2

Algol: fill(50,50,100,100,2);

Fortran: CALL FILL(50,50,100,100,2)

fill the block whose corners are (50,50) and (100,100) with intensity 2.

3.6 GLOAD (disc only)

Forms:
BASIC: CALL "GLOAD",A

Algol: gload(location(a[0]))

Fortran: CALL GLOAD(A)

Purpose: To load graphics memory from ordinary memory.

Remarks: GLOAD allows a whole picture to be loaded from data memory into the graphics board memory. A is the address in data memory at which the picture starts. In BASIC this is the address relative to the start of cache at which the picture starts (see Appendix D). In Algol and Fortran, A is usually the first element of an array which holds the picture.

The amount of data transferred depends on the current resolution, being 15360 bytes in high resolution and 7680 bytes in medium resolution. A medium resolution picture is loaded into the page currently selected by UPDATE.

Examples:

BASIC:
```
10 CLEAR 100,,120*128
20 CALL "RESOLUTION",0,2
30 ...read picture into cache
40 CALL "GLOAD",0
```

Algol:
```
BYTE ARRAY a[0:15359];
resolution(0,2);
 ...read picture into a;
gload(location(a[0]));
```

Fortran:
```
BYTE A(15360)
CALL RESOL(0,2)
 ...read picture into A
CALL GLOAD(A)
```

provide a skeleton for transferring a previously stored high resolution picture from disc back to the graphics board.

3.7 GSAVE (disc only)

Forms:

BASIC: `CALL "GSAVE",A`

Algol: `gsave(location(a[0]))`

Fortran: `CALL GSAVE(A)`

Purpose: To save the contents of graphics memory in ordinary memory.

Remarks: GSAVE allows a whole picture to be saved in data memory. A is the address in data memory at which the picture will start. In BASIC this is an address relative to the start of cache (see Appendix D). In Algol and Fortran, A is usually the first element of an array into which the picture will be copied.

The amount of data transferred depends on the current resolution, being 15360 bytes in high resolution and 7680 bytes in medium resolution. A medium resolution picture is copied from the page currently selected for UPDATE.

A call to GSAVE is normally followed by statements saving the picture in a file on disc.

Examples:

BASIC:
```
10 CLEAR 100,,60*128
20 CALL "RESOLUTION,1,4
30 ...generate picture
40 CALL "GSAVE",0
```

Algol:
```
BYTE ARRAY a[0:7679];
resolution(1,4);
 ...generate picture;
gsave(location(a[0]));
```

Fortran:
```
BYTE A(7680)
CALL RESOL(1,4)
 ...generate picture
CALL GSAVE(A)
```

provide a skeleton for transferring a medium resolution picture from the graphics board memory to data memory. The picture can then be saved on disc.

3.8 LINE

Forms:

BASIC:
```
CALL "LINE",X,Y,I
CALL "LINE",X,Y
```

Algol:
```
line(x,y,i)
```

Fortran:
```
CALL LINE(IX,IY,I)
```

Purpose: To draw a line on the screen of intensity I ending at the point (X,Y).

Remarks: LINE takes as its start point the end point of the previous line or the last plotted point, whichever occurred most recently, and draws a line to the end point (X,Y) specified in the call. X and Y can lie in the range -32768 to +32767; a line is only visible if it falls wholly or partially within the screen limits defined by RESOLUTION (and OFFSET in BASIC).

I determines the logical intensity of the line in the same way as for PLOT (see Section 3.11). If omitted (BASIC only), the value last specified in a call to PLOT, LINE or FILL is used.

CAUTION 1: the start point of the line is NOT plotted. (If it was, exclusive OR plotting would not work correctly.)

CAUTION 2: a line is usually not identical when drawn in the two possible directions. If you require to "unplot" a line by replotting it with background intensity or by exclusive OR plotting, make sure you plot it in the same direction as it was originally plotted.

Examples:

BASIC:
```
10 CALL "PLOT",0,0,3
20 CALL "LINE",100,50
```

Algol:
```
plot(0,0,3);
line(100,50,3);
```

Fortran:
```
CALL PLOT(0,0,3)
CALL LINE(100,50,3)
```

draw a line at intensity 3 from (0,0) to (100,50).

3.9 MIX (Algol and Fortran only)

Form:
Algol: i:=mix(r,g,b)

Fortran: I=MIX(IR,IG,IB)

Purpose: Returns an intensity value corresponding to the supplied values of the three primary colours red, green and blue.

Remarks: The integer procedure or function MIX provides an easy means for determining the intensity value I that corresponds to a desired combination of red, green and blue components passed as arguments r (IR), g (IG) and b (IB). r and g should lie in the range 0 to 7, and b in the range 0 to 3. The returned value I will lie in the range 0 to 255.

MIX is used to set up values in the byte arrays used in calls to COLOUR and VIEW when a colour system is available.

Examples:
Algol:
```
BYTE ARRAY a[0:15];
resolution(0,2);
a[0] = 0;
a[1] = mix(7,0,0);
a[2] = mix(0,7,0);
a[3] = mix(0,0,3);
colour(location(a[0]));
```

Fortran:
```
BYTE A(16)
CALL RESOL(0,2)
A(1)=0
A(2)=MIX(7,0,0)
A(3)=MIX(0,7,0)
A(4)=MIX(0,0,3)
CALL COLOUR(A)
```

set mode HR4 and the colour lookup table such that logical intensities 0, 1, 2 and 3 correspond to black, red, green and blue, respectively.

3.10 OFFSET (BASIC only)

Form:
BASIC: CALL "OFFSET",X,Y

Purpose: To change the XY coordinates of the lower left corner of the screen.

Remarks: X and Y can lie between -32768 and +32767. After a call to OFFSET the lower left corner of the screen corresponds to the point (X,Y).

CAUTION: graphics already drawn prior to a call to OFFSET are unchanged.

Example:
BASIC:
```
10 CALL "RESOLUTION",0,2
20 CALL "OFFSET",-160,-96
```

places the origin (0,0) in the centre of the screen.

3.11 PLOT

Forms:
BASIC: CALL "PLOT",X,Y,I
CALL "PLOT",X,Y

Algol: plot(x,y,i)

Fortran: CALL PLOT(IX,IY,I)

Purpose: To plot a point on the screen of intensity I at coordinates (X,Y).

Remarks: X and Y can lie in the range -32768 to +32767. However a point will only be displayed if its XY coordinates lie within the screen limits defined by RESOLUTION (and OFFSET in BASIC). The point (X,Y) is remembered and can be used as the start point of a line.

I determines the "logical" intensity of the point. Allowable values of I depend on the resolution and number of bits/pixel (see RESOLUTION). For example, in mode HR2 where 4 intensity values are available, the absolute value of I can range from 0 to 3.

If I is positive, the new intensity value replaces the old contents of the graphics memory at that point. If negative, the new value written to the graphics memory is the exclusive OR of the old value and the absolute value of I. A point plotted in this way is reversible by a further call to PLOT with the same negative value of I.

If I is 16, the XY coordinates are updated but no point is displayed. An intensity of 16 corresponds to a "pen up" move on a plotter.

If I is omitted (BASIC only) the value last specified in a call to PLOT, LINE or FILL is used. After RESOLUTION this value is zero.

Examples:
BASIC: 10 CALL "PLOT",100,50,2

Algol: plot(100,50,2);

Fortran: CALL PLOT(100,50,2)

plot a point of intensity 2 at (100,50).

3.12 RESOLUTION

Form:
BASIC: CALL "RESOLUTION",R,B

Algol: resolution(r,b)

Fortran: CALL RESOL(IR,IB)

Purpose: To select high or medium resolution and to set up the number of bits/pixel.

Remarks: RESOLUTION initialises the graphics hardware and clears the screen.

R may take the value 0 or 1, corresponding to high or medium resolution; B sets the number of bits/pixel. The following combinations of R and B are allowed:

R	B	Mode	Pages	Views	Intensities
0	2	HR2	1	1	4
0	1	HR1	1	2	2
1	4	MR4	2	1	16
1	2	MR2	2	2	4
1	1	MR1	2	4	2

where HR2 stands for "High Resolution, 2 bits/pixel" and so on, and page, view and intensity are defined at the beginning of this Chapter.

After a call to RESOLUTION the lower left corner of the screen corresponds to the XY coordinates (0,0). In high resolution modes the upper right corner is (318,191); in medium resolution modes it is (159,95). (These can be changed in BASIC by calling OFFSET.)

RESOLUTION initialises the "memorised" XY coordinates which are used as the start point of a line to (0,0).

Examples:
BASIC: 10 CALL "RESOLUTION",0,2

Algol: resolution(0,2);

Fortran: CALL RESOL(0,2)

set high resolution mode with 2 bits/pixel.

3.13 SETCOL (BASIC only)

Forms:

BASIC:
```
CALL "SETCOL",I,N
CALL "SETCOL",I,R,G,B
CALL "SETCOL"
```

Purpose: To modify the values held in the memory copy of the colour lookup table without affecting the table itself (see also VIEW and Appendix F).

Remarks: The SETCOL call sets up a value in the copy of the colour lookup table that is held in memory (in BASIC only). This copy is transferred to the hardware table by calling VIEW. This is in contrast to the COLOUR call which both modifies the copy of the table held in memory and loads the hardware table, assuming the page and view currently selected by DISPLAY.

SETCOL thus allows the display of non-standard combinations of views (for example, in HR1, of the differences between view 0 and view 1). It also allows the loading of the colour lookup table to be deferred until all colours have been set up.

I is the logical intensity to be modified and should be between 0 and 3 in high resolution and 0 and 15 in medium resolution.

The first form of the call with two arguments is used for black and white. N can range from 0 (black) to 255 (white). The second form with four arguments is for colour. R, G and B determine the red, green and blue brightness (R and G can be 0 to 7, B 0 to 3). The third form with no arguments restores the default values set up by RESOLUTION.

Example:

BASIC:
```
10 CALL "RESOLUTION",1,4
20 DATA ...specifying 16 colours
30 FOR I=0 TO 15
40 READ N
50 CALL "SETCOL",I,N
60 NEXT I
70 CALL "VIEW",0
```

sets mode MR4, then loads the colour lookup table to display page 0, the view being determined by the DATA statement.

3.14 UPDATE

Forms:

BASIC:
```
CALL "UPDATE",P,V     (all modes)
CALL "UPDATE",V       (HR 1)
CALL "UPDATE",P       (MR 4)
```

Algol: `update(p,v)`

Fortran: `CALL UPDATE(IP,IV)`

Purpose: To make page P, view V accessible for modification by PLOT, LINE or FILL.

Remarks: In mode HR2 there is only one page and view and this call is not used. In all MR modes there are two separate pages, numbered 0 and 1. In modes HR1 and MR2 there are two views, numbered 0 and 1, and in mode MR1 there are four views, numbered 0 to 3.

After a call to RESOLUTION, page 0 view 0 is selected for update, that is, accessible to PLOT, LINE and FILL. Calling UPDATE changes the page and/or view that is accessible. Similarly, calling DISPLAY selects the page and view that is displayed.

In BASIC only, P may be omitted in mode HR1 and V in mode MR4.

Examples:

BASIC:
```
10 CALL "RESOLUTION",1,1
20 CALL "UPDATE",1,3
```

Algol:
```
resolution(1,1);
update(1,3);
```

Fortran:
```
CALL RESOL(1,1)
CALL UPDATE(1,3)
```

set mode MR1, then select page 1, view 3 for update.

3.15 VIEW (BASIC)

Forms:
BASIC: CALL "VIEW",P (all modes)
CALL "VIEW" (HR only)

Purpose: To load the colour lookup table with values set up in memory by SETCOL, allowing the selection of non-standard views (see also SETCOL).

Remarks: The VIEW call copies the colour lookup table from memory to the actual table on the HRG board and selects an MR page. The table in memory is initialised by RESOLUTION and modified by COLOUR and SETCOL. Use of SETCOL followed by VIEW allows the selection of non-standard views; it also allows the loading of the hardware table to be deferred until all colours have been set up.

The argument P is the medium resolution page required and should be 0 or 1. It may be omitted in modes HR1 and HR2.

CAUTION: it is important to set up all four (HR) or sixteen (MR) intensity values by calling SETCOL before calling VIEW, otherwise undefined values will be loaded into the colour lookup table. This applies even in modes HR1 and MR1, and is in contrast to calling COLOUR in these modes, where only intensities 0 and 1 can be defined.

Example:
BASIC:
```
10 CALL "RESOLUTION",0,1
20 DATA 0,200,200,0
30 FOR I=0 TO 3
40 READ N
50 CALL "SETCOL",I,N
60 NEXT I
70 CALL "VIEW"
```

sets up HR1, then loads the colour lookup table to display the differences between view 0 and view 1.

3.16 VIEW (Algol, Fortran)

Form:
Algol: view(p,location(a[0]))

Fortran: CALL VIEW(IP,A)

Purpose: To load the colour lookup table directly from byte array a or A, and to select MR page p or IP (see also Appendix F).

Remarks: Like COLOUR, VIEW sets up values in the colour lookup table that determine the actual intensities or colours that correspond to the logical intensities plotted by PLOT, LINE and FILL. The difference is that with COLOUR only the first 2, 4 or 16 values are copied, depending on whether there are 1, 2 or 4 bits/pixel, whereas with VIEW the table is loaded directly with the first 4 (HR) or 16 (MR) values from the array. This allows the user to set up a non-standard view. For example, in HR1, the differences between view 0 and view 1 can be displayed.

The first argument, p (or IP), is the medium resolution page required and should be 0 or 1. It is ignored in high resolution. The second argument is the address of the byte array a (or A) whose contents have been previously set up. The array is normally of dimension 16 bytes to suit all resolutions. VIEW copies four or sixteen values from the array to the colour lookup table.

The value stored in each element can range from 0 (black) to 255 (white). For colour use the procedure (or function) MIX to combine the desired red, green and blue components into a single value.

Examples:
Algol:
```
BYTE ARRAY a[0:15];
resolution(0,2);
a[0] = 0; a[1] = 200;
a[2] = 200; a[3] = 0;
view(0,location(a[0]));
```

Fortran:
```
BYTE A(16)
DATA A/0,200,200,0,12*0/
CALL RESOL(0,2)
CALL VIEW(0,A)
```

set up HR1, then load the colour lookup table to display the differences between view 0 and view 1.

PAGE INTENTIONALLY BLANK

APPENDIX A

HRG BOARD INSTALLATION

The High Resolution Board on its own will only work with a black and white TV Monitor. An additional board is necessary for use with a TV Receiver via its aerial socket (black and white or colour) or for a Colour Monitor.

A.1 INSTALLATION INSTRUCTIONS

1 Open the 380Z case by removing the two screws from the rear of the case. Lift the lid up gently from the back then draw it backwards.

2 Identify the VDU board which is probably the second board from the right; it has a grey flat cable permanently attached to it.

3 Unplug the bus cable from the boards in the machine. The bus cable is the wide grey flat cable linking the boards together. (If your computer has only two boards it will not have a bus cable.) The bus cable connectors should be pulled gently upwards.

4 Unplug the connector leading from the keyboard; this is situated on the VDU board near the back of the computer.

5 Slide out the VDU board from its guides. You will notice that there is a small metal plug near the centre of the board. This is the video signal lead connector. Its lead is a twisted blue/yellow pair. There will also be a black lead plugged into the metal box. This is the TV signal lead. Unplug it from the box.

6 Hold the board flat and unplug the video lead from the VDU board. The plug is very tight, so take care not to jolt and damage the board when it comes unplugged.

7 Find an empty slot in the computer for the High Resolution Board to fit into. Make sure that one of the bus cable connectors is above it, and that there is still one above the rest of the boards in the machine.

8 Identify the empty IC socket on the VDU board. It looks like the sockets into which all the ICs on the boards are plugged, and is situated just below the attached flat grey VDU cable.

9 Plug the graphics board cable into the empty socket on the VDU board. The cable has a 16 pin plug on one end and a 14 pin plug on the other. The 16 pin end should go into the VDU board. You will see that the 16 pin plug has very few wires attached, and that they all enter from one side. The wires may fold back over the connector beneath a strain relief clamp. The GREEN wire should be the one towards the back of the machine when the plug is inserted into the socket on the VDU board. The YELLOW will then be towards the front of the machine.

Plug the TV cable connector back into the metal box. Slide the VDU board back into its slot.

10 Slide the High Resolution Graphics Board into the spare slot the same way round as the other boards.

11 Plug the graphics board cable into the graphics board. It plugs into the IC socket at the back, top edge of the board. The plug on the cable should have wires attached to all the pins. This (rainbow coloured) cable should be plugged in so that the BROWN strand is towards the back of the machine.

12 Take the blue/yellow video lead and plug it into the matching socket on the graphics board. It should click home.

13 Plug the bus cable back into the boards in the machine, and into the graphics board. The VDU board has no bus connector.

14 Reconnect the keyboard cable to the socket at the rear of the VDU board. The brown strand must be towards the rear of the machine. Check this.

15 Reconnect the VDU board to the CPU board by means of the grey cable permanently attached to the VDU board.

16 Check that there are no spare bits left over and that there are no plugs out of their sockets. Have you plugged in the graphics board cable correctly? (See 9 and 11). Have you plugged in the keyboard cable correctly? (See 14).

17 Switch on the machine. The COS prompt should appear on the screen. If it does not, switch off immediately and refer to Section A.3.

18 Enter the test program in Chapter 2, Section 2.1, into the computer. If that doesn't work, refer to Section A.3.

19 Replace the lid of the computer.

A.2 INSTALLATION SUMMARY

Plug the 16 pin end of the graphics cable into the VDU board and the 14 pin end into the graphics board which should be slid into the computer in a spare slot. Plug the bus cable into the graphics board. Unplug the video lead from the VDU board and plug it into the video socket on the graphics board. The video output from the computer then consists of normal VDU output mixed with graphics output.

A.3 WHAT TO DO IF IT DOESN'T WORK

The following points should help if the system appears not to work once the graphics board has been installed:

1 Is the light above the power switch on?

2 Is the power cable plugged into the CPU board?

3 Is the blue/yellow video lead plugged into the graphics board?

4 Is the lead from the VDU to the graphics board plugged in the correct way round and the correct way up? Brown towards the back at the graphics end? Single wire towards the top at the VDU end?

5 Is the monitor on?

If the picture is unstable, check the following points:

1 Check the cable from VDU to graphics board. The VDU to graphics board cable may be loosely plugged in or have a wire loose, particularly at the VDU end.

2 Check that the monitor hold controls have not been disturbed. (They should need no adjustment).

If these suggestions do not help and you cannot get a picture, unplug the video lead from the graphics board and plug it back into the similar socket on the VDU board to return the system to normal. The monitor should now display only VDU output, direct from the VDU board, and should help to determine whether the malfunction is due to something you have done during installation or to a non-functioning graphics board.

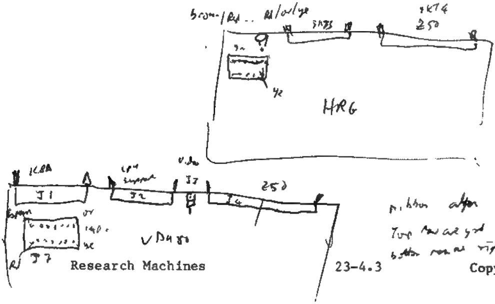
HRG - VDU-80

Link cable is 14 pin both ends HRG, VDU-80

on HRG end. or + ye are cut at the Hdr

The or + ye from the ribbon are terminated in a mini coax plug this goes into the video (HRG) o/p on Top of the cable next to the 14 pin DIL SKT11 ... cable ... folded over ... strain relief

The top row are gnd
bottom row are signal



ribbon after brown/red/or — red/or/ye
Top row are gnd
bottom row are signal

PAGE INTENTIONALLY BLANK

APPENDIX B

EXAMPLE PROGRAMS

This Appendix contains a number of BASIC programs that generate pictures. Their main purpose is to illustrate various aspects of the graphics support routines; they are not intended to perform useful functions. However they do embody a few sections which may be of use in other programs.

B.1 STAR

This program illustrates line drawing in high resolution. The pictures are drawn with 2 bits/pixel but since only intensity 3 is used, they could equally well be in mode HR1. OFFSET is used to simplify the calls to PLOT and LINE.

```
100 REM-- STAR
110 DIM X(15),Y(15)
120 R1=110: REM X RADIUS
130 R2=90: REM Y RADIUS
140 X0=159: REM X CENTRE
150 Y0=95: REM Y CENTRE
160 GRAPH 1: GRAPH 0
170 P2=ATN(1)*8
180 Z=3

190 FOR N=3 TO 15   for each star
200 A1=P2/N

210 FOR J=1 TO N   compute coordinates
220 A=A1*J
230 X(J)=R1*COS(A)
240 Y(J)=R2*SIN(A)
250 NEXT J

260 CALL "RESOLUTION",0,2   draw star
270 CALL "OFFSET",-X0,-Y0
280 FOR J=1 TO N-1
290 FOR K=1 TO N-J
300 CALL "PLOT",X(K),Y(K),Z
310 CALL "LINE",X(K+J),Y(K+J)
320 NEXT K
330 NEXT J

340 REM-- WAIT A WHILE
350 FOR J=0 TO 1000: NEXT J
360 NEXT N
370 GOTO 190
```

B.2 COSINE HISTOGRAM

This program shows how to draw a histogram of data which in real life

might be the frequency distibution of events, etc. The histogram is drawn in two tones to make it more dramatic (remove line 180 to keep to one tone).

Lines 240 onwards are a simple general purpose axis routine where XL and XH are the absolute X coordinates of the ends of the X axis, XY is its Y coordinate, XI is the X axis increment and TI is the size of the tick marks. YL, YH, YX and YI are the corresponding Y axis values

```
100 REM-- COSINE HISTOGRAM
110 GRAPH 1:GRAPH 0
120 CALL "RESOLUTION",0,2
130 A=8*ATN(1)/320
140 Z=0

150 FOR J=0 TO 310 STEP 10
160 Y=80*(1-COS(A*(J+5)))
170 CALL "FILL",J,0,J+9,Y,Z+1
180 Z=1-Z
190 NEXT J

200 XL=0:XH=318:XY=0:XI=10
210 YL=0:YH=191:YX=160:YI=10:TI=2
220 GOSUB 240
230 END

240 REM-- DRAW AXES
250 CALL "PLOT",XL,XY,3
260 CALL "LINE",XH,XY
270 FOR X=XL TO XH STEP XI
280 CALL "PLOT",X,XY-TI,3
290 CALL "LINE",X,XY+TI
300 NEXT X
310 CALL "PLOT",YX,YL,3
320 CALL "LINE",YX,YH
330 FOR Y=YL TO YH STEP YI
340 CALL "PLOT",YX-TI,Y,3
350 CALL "LINE",YX+TI,Y
360 NEXT Y
370 RETURN
```

B.3 CUBISM

This short program draws cubist pictures. It demonstrates the speed of the block fill and provides a trivial example of "XOR plotting" (variable C may be negative). A more serious use of XOR plotting is in "undrawing" points and lines by plotting the same data twice with the identical negative intensity.

```
100 REM-- CUBISM
110 GRAPH1: GRAPH 0
120 RANDOMIZE
130 CALL "RESOLUTION",1,4
```



```
140 X0=RND(1)*160
150 X1=RND(1)*160
160 IF X0>X1 THEN T=X0: X0=X1: X1=T
170 Y0=RND(1)*96
180 Y1=RND(1)*96
190 IF Y0>Y1 THEN T=Y0: Y0=Y1: Y1=T
200 C=INT(RND(1)*31-15)
210 CALL "FILL",X0,Y0,X1,Y1,C
220 GOTO 140
```

B.4 REVOLVE

This program shows how you can draw up to 8 pictures in separate pages and views (selected by variables P and V), then display them in sequence to give an impression of movement. In order to make the program as clear as possible, the pictures have been kept very simple.

```
100 REM-- REVOLUTION
110 CALL "RESOLUTION",1,1
120 GRAPH 1:GRAPH 0

130 REM-- CREATE 8 PICTURES
140 P1=ATN(1)*4
150 X0=80
160 Y0=42
170 R=40
180 A=P1/8
190 FOR P=0 TO 1
200 FOR V=0 TO 3
210 CALL "UPDATE",P,V
220 A1=A*(4*P+V)
230 X=R*COS(A1)+X0
240 Y=R*SIN(A1)+Y0
250 CALL "PLOT",X,Y,1
260 X=R*COS(A1+P1)+X0
270 Y=R*SIN(A1+P1)+Y0
280 CALL "LINE",X,Y,1
290 CALL "DISPLAY",P,V
300 NEXT V
310 NEXT P

320 REM-- DISPLAY EACH IN TURN
330 FOR P=0 TO 1
340 FOR V=0 TO 3
350 CALL "DISPLAY",P,V
355 REM-- WAIT A WHILE
360 FOR I=1 TO 10
370 NEXT I
380 NEXT V
390 NEXT P
400 GOTO 330
```

B.5 GRAPH

This program plots a series of graphs of random data on a grey grid. Block fill is used to plot a little square at each graph point; the points are joined by lines. High resolution with one bit/pixel is used, giving two views. The grid is plotted as view 0 and its intensity is set to 128 by line 140 (it would be 200 by default). The graphs are drawn as view 1 but the SETCOL and VIEW calls are used to load the colour lookup table directly, thus allowing view 0 and view 1 to be displayed simultaneously. Intensities 2 and 3 govern the intensity of the graph; this is set to 255 by lines 260-280. At line 440 the graph is made invisible by setting intensity 2 to 0 and intensity 3 to the grey grid colour (to deal with graph points on the grid). Then the graph is erased by calling CLEAR. Note that the grid remains visible throughout.

This is a fairly advanced example illustrating direct loading of the colour lookup table. Try changing line 450 to

```
450 DATA 0,128,0,0
```

How do you account for what happens? (See Section 2.16.)

```
100 REM-- GRAPH DRAWING
110 GRAPH 1:GRAPH 0
120 CALL "RESOLUTION",0,1

130 REM-- MAKE GRID
140 CALL "COLOUR",1,128
150 FOR X=0 TO 310 STEP 10
160 CALL "PLOT",X,0,1
170 CALL "LINE",X,190
180 NEXT X
190 FOR Y=0 TO 191 STEP 10
200 CALL "PLOT",0,Y,1
210 CALL "LINE",310,Y
220 NEXT Y

230 CALL "UPDATE",1
240 REM-- LOOP TO PLOT GRAPHS
250 REM-- DISPLAY VIEW 0 AND VIEW 1
260 DATA 0,128,255,255
270 RESTORE 260
280 GOSUB 510
290 I=20:J=2
300 REM-- DO 1ST POINT
310 X0=0:Y0=RND(1)*190
320 CALL"FILL",X0-J,Y0-J,X0+J,Y0+J,1
330 REM-- DO REST OF GRAPH
340 FOR X=I TO 310 STEP I
350 Y=RND(1)*190
360 CALL"PLOT",X0,Y0,1
370 CALL"LINE",X,Y
380 CALL"FILL",X-J,Y-J,X+J,Y+J
390 X0=X:Y0=Y
400 NEXT X

410 REM-- WAIT A WHILE
420 FOR K=0 TO 4000
```



```
430 NEXT K
440 REM-- BLANK VIEW 1
450 DATA 0,128,0,128
460 RESTORE 450
470 GOSUB 510
480 REM-- CLEAR VIEW 1 AND LOOP BACK
490 CALL "CLEAR"
500 GOTO 270

510 REM-- SET COLOUR VECTOR FROM DATA
520 FOR K=0 TO 3
530 READ N
540 CALL "SETCOL",K,N
550 NEXT
560 CALL "VIEW"
570 RETURN
```

The effect of a static grid with changing graphs can also be obtained in mode HR2 by plotting the graph in XOR mode but the graph coordinates would have to be stored to allow them to be used again for "unplotting". Try it!

PAGE INTENTIONALLY BLANK

APPENDIX C

ASSEMBLY LANGUAGE PROGRAMMING

C.1 BASIC ADDRESSING

The high resolution board contains 16K bytes of graphics memory. Like the memory mapped VDU, this block of memory is normally connected to its video circuitry and is not accessible on the Z80 bus. When it is necessary to modify the video memory contents, the memory is "opened", modified, and "closed". As with the VDU, the video output goes blank while the memory is open and it is necessary to open it only during line or frame blanking to avoid flicker. The details are given later.

The graphics memory base address is at 0F000 hex. To allow access to all 16K, it is divided into twelve "windows" or pages, where the page address is determined by the more significant part of the Y address. Page addresses range from 0 to B hex; each page consists of 1280 bytes, addressed from 0F000 to 0F4FF hex.

There are two control ports, Port 0 (read/write) at FB00 and Port 1 (write only) at FB01.

C.2 HIGH AND MEDIUM RESOLUTION

The board can operate in two basic modes, giving "high" and "medium" resolution, respectively. These complement the "low" resolution graphics of the standard 380Z. In both cases, the graphics area is rectangular, the vertical extent overlapping the upper twenty lines of text on the VDU and the horizontal extent being similar to the VDU width (40 characters). It thus matches the "graphics window" available in BASIC after "GRAPH 1" and addressed by the "PLOT" keyword.

In high resolution mode, the bottom left corner is (0,0) and the top right (319,191). Each point can have one of four intensities. In medium resolution mode the corners are (0,0) and (159,95) and sixteen intensities are available. Because of hardware restrictions, the extreme right hand column (X = 319) is displayed at reduced intensity and is best avoided.

The actual intensity that appears on the screen is preset in a 16 byte scratchpad memory. At display time the "logical" intensity stored in the picture is mapped to a "physical" intensity stored in the scratchpad. With a colour monitor, the various physical intensities (which can range from 0 to 255) map to different colours.

Since the scratchpad content can be changed rapidly, it is easy to turn pictures off (set physical intensity to zero), fade them up or down (increment or decrement physical intensity) and select portions (write segments of pictures at different logical intensities, then manipulate the corresponding physical ones). In addition, medium resolution uses only half the video memory and two separate "pages" are available, by setting bits in Port 0.

C.3 HIGH RESOLUTION

The primary feature which determines that the board will display in high resolution (320 by 192) is the contents of Port 0; the three high order bits of the Port must be zero. The five low order bits are concerned with opening video memory and writing to the scratchpad RAM and should be preset to 00011B. Thus the initialisation byte for high resolution for Port 0 is 00000011B. Although Port 0 can be read, what is read is not directly related to what is written and a "mask" (a readable record) should be maintained in user RAM, from which Port 0 is copied when required. (COS uses a similar method to keep track of the Port 0 380Z control port. Refer to the COS Monitor listing if you find the concept difficult.)

Once Port 0 has been initialised, each video byte represents four points, each two bits specifying a logical intensity level in the range 0 to 3. By convention, the scratchpad RAM is loaded with physical intensities such that 0 is black and 3 white (scratchpad contents of 0,64,128,255 gives nicely graded shades from black to white, corresponding to logical intensities 0,1,2,3). Representing the selected logical intensity as "ij", the bits of a high resolution video byte can be represented as "ijijijij", each "ij" therefore being one pixel. On the above convention for physical intensity:

```
00000011   white dot at x=0
00000100   dark grey dot at x=1
00100000   light grey dot at x=2
11001100   white dots at x=1 and 3
```

where "x" is relative to the current X position modulo 4.

Apart from the bit pair within a byte, which as we have just seen is determined by the least significant two bits of the X coordinate, the XY coordinates determine the address in video RAM, as follows. The bottom four bits of the video address are the four less significant Y bits. Above these are the 7 remaining X bits. At the top is the video "base" address, 0F000 hex. The 4 more significant Y bits determine the video "page" and are written to Port 1 before accessing video memory. This is best seen in a diagram:

```
Port 1 data     0000 YYYY
                     7654

                ---------
Video address 1111 XXXX XXXY YYYY
                   876 5432 3210
```

(top 4 bits from base addr 0F000)

It should be noted that the physical Y address determined thus is inverted. To map the logical Y address (passed as a parameter) to the physical address it must be subtracted from 191. The process of plotting a single high resolution point can now be summarised:

a. Get x,y,i parameters and check that they are in range. Store x,y for later use (e.g. start point of a line).

b. Form intensity "mask". Given that specified (logical) intensity is ij, form the byte ijijijijB.

c. Form video address from base address (0F000), x bits 2 to 8 and y bits 0 to 3. Write y bits 4 to 8 in lower four bits of Port 1.

d. Form masks for the read-modify-write of the video byte, using x bits 0 and 1. The AND mask to isolate the wanted pixel will be 11111100, 11110011, 11001111 or 00111111. The OR mask will be the previously constructed intensity word ANDed with the 1's complement of the AND mask.

e. Open video RAM in frame or line blanking, read contents of selected address, AND with mask, OR with mask and write back.

C.4 ANOTHER HR MODE

It is evident that it would be possible to regard HR mode as having two "views", each of which had a resolution of 320x192 if the intensities were manipulated appropriately. Thus if two pictures were drawn, one at intensity 1 (01B) and the other at intensity 2 (10B), the two could be shown separately by setting the scratchpad to 0,255,0,0 or to 0,0,255,0. However a problem arises when the same point is present in both pictures. Setting the point to intensity 2 will reset the point on picture 1 (it was 01B and has been set to 10B). The opposite also occurs if the intensity 1 picture is drawn second.

This problem can be overcome by a little juggling. If the two intensities (let them be B and F for off and on or background and foreground) for each page are stored, the page to display can be set by loading the scratchpad with B,F,B,F or B,B,F,F. For page zero, the AND mask would be 11111110, 11111011, 11101111 or 10111111. The OR mask would be the 1's complement. The intensity mask would be simply 11111111. For page one, the AND mask would be 11111101, 11110111, 11011111 or 01111111.

Hence before calling any routines which would write to video RAM, a call would be made to a routine to set up the "view" to write to (0 or 1), which would in turn set up the appropriate masks.

In summary, in high resolution two modes are possible, both with a resolution of 320 by 192. In the first there is one page with two intensity bits; in the second there are two pages but only one intensity bit.

C.5 MEDIUM RESOLUTION

Medium resolution mode is set by initialising Port 0 to A3 hex (page 0) or C3 hex (page 1). This gives X and Y ranges of 0 to 159 and 0 to 95, respectively, with 16 intensities. Each video byte corresponds to two points (i.e. there are 4 bits/pixel). Assuming the selected intensity is "ijkl", the video byte can be represented as "ijijklkl", so that:


```
00110011   white dot at x=0
00000100   very dim dot at x=1
```

where x is relative to the current X position modulo 2, and there are 4 bits per pixel.

The video byte is constructed from the three less significant Y bits, all the X bits except the least significant and the base address, as follows:

```
Port 1 data     0000 YYYY
                     6543

Video address 1111 0XXX XXXX YYYP
                    765 4321 210

        (top 4 bits from base addr 0F000)
```

with the least significant bit of the X coordinate determining which half of the video byte to use. "P" is the page bit. The procedure for plotting is similar to high resolution except that the masks are different, and, because of the availability of two pages, only the even or odd addresses are used, depending on the page.

C.6 OTHER MR MODES

As with HR, the other modes of 4 pages with 2 intensity bits and 8 pages with 1 intensity bit can be achieved by juggling, in addition to the mode with 2 pages and 4 intensity bits described above.

C.7 SETTING THE INTENSITY OR COLOUR

The scratchpad content is always 16 bytes. For medium resolution with sixteen intensities, the values in the scratchpad correspond to the values stored in video RAM on a one to one basis. For high resolution with four intensities, the values are repeated four times. Thus for the example mentioned above in Section C.3, the sequence 0,64,128,255 would be stored four times. The scratchpad should be filled during frame blanking to avoid flicker; this is indicated by bit 0 of Port 0 going low. To ensure that the start of frame blanking is found, one should wait for the bit to go high, then low.

The scratchpad is then open and can be written to by first placing the colour address in the four high bits of Port 1 with intensity data on the four low bits. Since the intensity data byte is 8 bits wide, it must be written to the scratchpad in two 4 bit nibbles:

```
Port 1 data   AAAA DDDD   (low nibble)
              3210 3210

              AAAA DDDD   (high nibble)
              3210 7654
```

Toggling bit 0 of Port 0 low, then high writes the low nibble to the scratchpad. Toggling bit 1 writes the high nibble. Each nibble is

written separately.

When a colour converter is in use (RGB or PAL), 3 of the 8 intensity bits are assigned to red, 3 to green and 2 to blue, giving possible ranges for red, green and blue of 0 to 7, 0 to 7 and 0 to 3, respectively. The bits are arranged in the intensity byte as

```
GRGBRGBR
22111000
```

where R0 is the least significant red bit, R2 the most significant, and so on.

Some useful values are:

	Hex	Dec
White	FF	255
Cyan	B6	182
Magenta	5B	91
Yellow	ED	237
Grey	D0	208

Graded shades of red, green and blue are given by the hex values:

	Bright						Dim
Red	49	48	41	40	9	8	1
Green	A4	A0	84	80	24	20	4
Blue	12			10			2

C.8 ACCESSING VIDEO RAM

Video RAM is normally closed, i.e. not accessible to the CPU. By setting the "Open Request" bit (ORQ) in Port 0 it is automatically opened during line and frame blanking. (This is more efficient than the low resolution VDU, in which it is necessary to set up the open state each time.) A routine which needs to access video RAM need only set ORQ once; it then can make as many (synchronised) accesses as it likes, finally closing by resetting ORQ before exit.

One penalty of this is the clash that occurs if debugging of such routines is attempted with the "Front Panel". After ORQ is set, both HRG RAM and VDU RAM may be open at the same address and can clash on the data bus if single stepping or break pointing is attempted. The Front Panel screen will fill with garbage and HRG RAM will be corrupted. The situation is best avoided as it could cause permanent damage to the hardware.

The ORQ bit is bit 2 of Port 0; it can be set or cleared at any time. Once set, the procedure for synchronising to blanking prior to accessing video RAM is:


```
PORT0   EQU     0FB00H
FRAME   EQU     0
LINE    EQU     1

        LD      HL,PORT0
        BIT     FRAME,(HL)
        JR      Z,W3
W1:     BIT     LINE,(HL)
        JR      Z,W1
W2:     BIT     LINE,(HL)
        JR      NZ,W2
W3:     NOP
```

Note the NOP instruction at W3 which allows 1.25 usec for the open state to establish itself. After synchronisation has been established there is time for a read modify write sequence such as:

```
LD      A,(DE)  ;READ
AND     B       ;MODIFY
OR      C
LD      (DE),A  ;WRITE
```

before the end of line blanking when the video RAM automatically closes again.

C.9 CLEARING THE SCREEN

In some circumstances it is useful to open video RAM continuously (which of course blanks the display) and an example of this is when clearing the screen to zero during initialisation. An OPEN bit, bit 3, in Port 0 causes this state; it should only be set during frame blanking. ORQ must also be set.

C.10 USE AS MEMORY

In 32K systems video RAM can be used as additional memory when not in use for graphics; it occupies the 16K segment with addresses from 7C00 to BBFF inclusive. To switch from graphics to memory usage, bit 4 of Port 0 should be set.

For 56K systems MB can replace Mapping PROM with which disables M command from mapping HRG RAM 6 7C00 - BBFF

APPENDIX D

SAVING AND LOADING PICTURES ON DISC

This section, covering how to save and load pictures, has been placed in an Appendix because it is at present applicable only to the disc environment and because the mechanism may change in future releases of the graphics support package.

The descriptions below apply to the BASIC implementation; however the same general principles apply in Algol and Fortran. In these languages, however, the "cache" memory required would be declared within the program as a byte array, and cache data would be transferred to and from disc using the built in facilities of the language.

D.1 SAVING A PICTURE

Saving a picture is a two stage process. The picture is first transferred from graphics memory (on the HR board) to a previously reserved area of main memory (called the "cache") by a call to GSAVE. Then the saved picture is written out to disc by a call to WRITE. Prior to this, a data file must have been successfully opened for writing, by executing a BASIC CREATE statement (formerly FILES 2). After the picture has been written out to disc the file is closed by a CLOSE statement.

In high resolution the whole of graphics memory is saved, comprising 120 disc sectors of 128 bytes each (about 15K bytes). No distinction is made between mode HR2 (2 bits/pixel) and mode HR1 (2 views each with 1 bit/pixel). In HR1 the two views are saved together. In medium resolution, only the page currently selected for UPDATE is saved so the amount of data is halved; the storage required is 60 disc sectors (about 7.5K bytes). Again, the whole of a page is saved; this consists of 1, 2 or 4 views, depending on whether the mode is MR4, MR2 or MR1.

Cache storage is reserved by a modified form of the CLEAR statement. A third argument specifies the number of bytes to reserve. For example, the statement

```
10 CLEAR 100,,120*128
```

sets up a cache area equivalent to 120 sectors (15360 bytes), enough to store one high resolution picture. (The second argument, whose presence is implied by the paired commas, is not yet implemented but will later be used to reserve space for multiple data files.) Remember that CLEAR clears all variables and arrays and that the first argument (100 in the example) sets the size of string space; it is recommended that CLEAR be the first statement of a program.

A call to GSAVE is of the form

```
CALL "GSAVE",A
```

where A is the address in cache, in bytes relative to the start of cache, at which the saved graphics image will begin. Thus CALL "GSAVE",0 saves

the current picture in cache, starting at the first available address, and if the cache area reserved by CLEAR is the same size as the graphics memory (as it is in the example above of the use of the CLEAR statement) the picture will just fit. The A argument allows more than one picture to occupy cache if enough memory is available.

A call to write is of the form

```
CALL "WRITE",F,R,A,N
```

where F is the file number (currently only 10 is valid), R is the record number in the file at which to start saving the picture (expressed in sectors, starting from zero, and relative to the start of the file), A is the byte address relative to the start of cache from which to start saving and N is the number of sectors to write. (If omitted, N is assumed to be 1.)

This may seem rather complicated but the procedure for saving a picture is in fact quite straightforward as the following examples show:

Example 1 - High Resolution

```
10 CLEAR 100,,120*128
20 CALL "RESOLUTION",0,2
30 REM generate picture...
90 REM ...then save it
100 CALL "GSAVE",0
110 CREATE #10,"PIC1.PIC"
120 CALL "WRITE",10,0,0,120
130 CLOSE #10
```

Example 2 - Medium Resolution

```
10 CLEAR 100,,60*128
20 CALL "RESOLUTION",1,4
30 REM generate picture...
90 REM ...then save it
100 CALL "GSAVE",0
110 CREATE #10,"PIC2.PIC"
120 CALL "WRITE",10,0,0,60
130 CLOSE #10
```

As can be seen, the only differences between high and medium resolution are the differing sizes involved and the possible need to call UPDATE (not shown in Example 2) to select the appropriate page before calling GSAVE.

D.2 LOADING A PICTURE

Loading a picture back into graphics memory from disc is essentially the opposite of saving it. The file containing the picture is first opened by means of the OPEN keyword (formerly FILES 1). The picture is read into cache memory by a call to READ and then transferred to graphics memory by a call to GLOAD. The form of a call to READ is similar to that

for WRITE and is

```
CALL "READ",F,R,A,N
```

where F is the file number (restricted to 10 for now), R is the record number at which the picture starts in the file (in sectors, starting from zero), A is the first cache address to use (in bytes relative to the start of cache and starting from zero) and N is the number of sectors to transfer (defaulting to 1 if omitted). The form of a call to GLOAD is similar to that for GSAVE and is

```
CALL "GLOAD",A
```

where A is the cache address where the picture starts, in bytes relative to the beginning of cache.

Example 3 - High Resolution

```
10 CLEAR 100,,120*128
20 CALL "RESOLUTION",0,2
30 OPEN #10,"PIC1.PIC"
40 CALL "READ",10,0,0,120
50 CALL "GLOAD",0
```

Example 4 - Medium Resolution

```
10 CLEAR 100,,60*128
20 CALL "RESOLUTION",1,1
30 OPEN #10,"PIC3.PIC"
40 CALL "READ",10,0,0,60
50 CALL "UPDATE",1,0
60 CALL "GLOAD",0
```

In Example 4, we have set up mode MR1 with 4 views in each page; line 50 causes GLOAD to transfer 4 views from PIC3 into page 1. These could then be displayed in turn by calls to DISPLAY.

D.3 SAVING THE COLOUR TABLE

Sometimes it is desirable to save the contents of the colour lookup table along with the picture, particularly in medium resolution where there may be 16 special settings each with a red, green and blue value. Since the contents of the colour table cannot be read back from the graphics board it is necessary to maintain a copy of it in memory and in the following program fragments this is kept in the two dimensional array C. For the purposes of these examples it is assumed that medium resolution is being used along with colour graphics.

A convenient place to store the colour vector is in the first record of the picture file (record 0), with the graphics image following (records 1 to 60). At the beginning of the program we place the statements

```
10 CLEAR 100,,61*128
20 DIMENSION C(15,2)
```



```
30 CALL "RESOLUTION",1,4
```

to reserve the necessary storage, etc. Note the increase in cache size, compared with Examples 2 and 4. When the time comes to save the picture, we first use the following statements to move the colour information into the first 128 bytes of cache:

```
100 FOR I=0 TO 15
110 FOR J=0 TO 2
120 CALL "STOREB",C(I,J),I*3+J
130 NEXT J
140 NEXT I
```

This transfers 48 bytes, using STOREB whose syntax is

```
CALL "STOREB",V,A
```

where V is the value to store (range -128 to +255) and A is the cache address in which to store it. As usual A is in bytes relative to the start of cache and starts from zero. Next the picture is transferred to cache, starting at address 128 and then the colour vector and picture are written out to disc:

```
150 CALL "GSAVE",128
160 CREATE #10,"PIC1V.PIC"
170 CALL "WRITE",10,0,0,61
180 CLOSE #10
```

The file is one sector longer than that generated by Example 2 because of the colour vector. (It is not necessary to align the picture with a record boundary; it could have been copied to cache with CALL "GSAVE",48 saving a few bytes. However starting the graphics information at the beginning of a record makes it easier to read back separately, should this be required.)

As before, reading back the colour information along with the picture is essentially the reverse process:

```
10 CLEAR 100,,61*128
20 DIMENSION C(15,2)
30 CALL "RESOLUTION",1,4
40 OPEN #10,"PIC1V.PIC"
50 CALL "READ",10,0,0,61
60 FOR I=0 TO 15
70 FOR J=0 TO 2
80 CALL "RECALLB",VARADR(C(I,J)),I*3+J
90 NEXT J
100 NEXT I
110 FOR I=0 TO 15
120 CALL "SETCOL",I,C(I,0),C(I,1),C(I,2)
130 NEXT I
140 CALL "GLOAD",128
150 CALL "VIEW",0
```

This example uses a call to RECALLB at line 80; as you might expect, this recalls a byte from cache into a numeric variable or array element,

performing the inverse function to STOREB. It is important to use the exact syntax for this call, which is

```
CALL "RECALLB",VARADR(V),A
```

where V is a numeric variable (or an element of an array) which will receive a value and A is the cache address of the value to recall, in bytes relative to the start of cache. The use of the function VARADR is essential. It causes the address of V to be passed to RECALLB, rather than its value. It is necessary for RECALLB to know the address of V in order to transfer the specified value there. Some checking is done to ensure that the address is reasonable but it is not possible to be absolutely sure that the user has passed an address (via VARADR) rather than a value. Please be careful then, when using RECALLB, always to use VARADR.

D.4 CONCLUSION

This Appendix has presented sufficient information to allow the user to save and recall graphics pictures on disc. The methods described can be elaborated in a number of ways to allow libraries of pictures to be maintained and recalled selectively and this use is entirely legitimate.

It may occur to users, however, that the calls described in this Appendix could also be used as the basis of a general method of random access to files. This is indeed the case, but such usage is discouraged. It is planned to provide a much simpler method of random file access in a future release of RML BASIC.

PAGE INTENTIONALLY BLANK

APPENDIX E

HRG QUICK REFERENCE GUIDE (BASIC)

This Appendix summarises the BASIC graphics support and can be kept by the computer for reference. For more detail refer to Chapter 3.

```
CALL "RESOLUTION",R,B
```

Must be first call to graphics routines and initialises the system. R defines resolution to be used - if R=0 then high resolution (319x192) else if R=1 then medium resolution (160x96). B is number of bits per pixel and can be 1 or 2 in HR, or 1, 2, or 4 in MR. If B is less than the maximum, multiple views of the graphics memory are possible.

```
CALL "PLOT",X,Y[,I]
```

Plot a point at coordinates X,Y in current page and view. I is the logical brightness or colour of the plotted point. The range of possible values depends on the number of bits/pixel - 0-1 if 1 bit/pixel, 0-3 if 2 bits/pixel, or 0-15 if 4 bits/pixel. If I is less than zero, its absolute value is exclusive ORed (XOR) with the old contents of the pixel. If it is greater than 15, no point is plotted, but the "pen" (see LINE) is moved to X,Y. I defaults to the last value specified.

```
CALL "LINE",X,Y[,I]
```

Draw a line from the current "pen" position to X,Y. I has the same meaning as for PLOT. The "pen" position is the last value of X,Y given in a PLOT or LINE call.

```
CALL "FILL",X1,Y1,X2,Y2[,I]
```

Fill the rectangle specified by the points X1,Y1 and X2,Y2. I has the same meaning as in PLOT and LINE. X2 must be greater than X1 and Y2 greater than Y1.

```
CALL "COLOUR",I,N
CALL "COLOUR",I,R,G,B
CALL "COLOUR"
```

The logical colour I takes the physical brightness N or colours R, G and B. I can be in the range 0-15 and N in the range 0-255. R and G (red and green) can be 0-7 and B (blue) 0-3. If all arguments are omitted, the default settings are restored.

APPENDIX E

HRG QUICK REFERENCE GUIDE (BASIC)

This Appendix summarises the BASIC graphics support and can be kept by the computer for reference. For more detail refer to Chapter 3.

```
CALL "RESOLUTION",R,B
```

Must be first call to graphics routines and initialises the system. R defines resolution to be used - if R=0 then high resolution (319x192) else if R=1 then medium resolution (160x96). B is number of bits per pixel and can be 1 or 2 in HR, or 1, 2, or 4 in MR. If B is less than the maximum, multiple views of the graphics memory are possible.

```
CALL "PLOT",X,Y[,I]
```

Plot a point at coordinates X,Y in current page and view. I is the logical brightness or colour of the plotted point. The range of possible values depends on the number of bits/pixel - 0-1 if 1 bit/pixel, 0-3 if 2 bits/pixel, or 0-15 if 4 bits/pixel. If I is less than zero, its absolute value is exclusive ORed (XOR) with the old contents of the pixel. If it is greater than 15, no point is plotted, but the "pen" (see LINE) is moved to X,Y. I defaults to the last value specified.

```
CALL "LINE",X,Y[,I]
```

Draw a line from the current "pen" position to X,Y. I has the same meaning as for PLOT. The "pen" position is the last value of X,Y given in a PLOT or LINE call.

```
CALL "FILL",X1,Y1,X2,Y2[,I]
```

Fill the rectangle specified by the points X1,Y1 and X2,Y2. I has the same meaning as in PLOT and LINE. X2 must be greater than X1 and Y2 greater than Y1.

```
CALL "COLOUR",I,N
CALL "COLOUR",I,R,G,B
CALL "COLOUR"
```

The logical colour I takes the physical brightness N or colours R, G and B. I can be in the range 0-15 and N in the range 0-255. R and G (red and green) can be 0-7 and B (blue) 0-3. If all arguments are omitted, the default settings are restored.


```
CALL "SETCOL",I,N
CALL "SETCOL",I,R,G,B
CALL "SETCOL"
```

Similar to COLOUR, except that the effect of this call is delayed until a call to VIEW (q.v.) is made. It enables the setting up of non-standard views and allows a very rapid change of all of the colours.

```
CALL "VIEW"[,P]
```

Transfer the colour changes as specified by SETCOL to the colour lookup table. P (which can be omitted in HR mode) specifies the MR page on which to effect the changes.

```
CALL "UPDATE"[,P],V
```

In MR, there are two pages of memory, which can be written to or displayed independently. P specifies the page which will be written to by subsequent calls to PLOT, LINE, and FILL. It can be omitted in HR mode. In a similar manner, there can be two or four views, or logical pages of memory, which can be written to or displayed independently, if the number of bits/pixel is less than the maximum. This can occur in both HR and MR. V specifies the view to be written to by subsequent calls to PLOT, LINE and FILL.

```
CALL "DISPLAY"[,P],V
```

Specify which page and view is to be displayed. As in UPDATE, P can be omitted in HR mode.

```
CALL "CLEAR"
```

Clear the current page and view, filling it with logical intensity 0.

```
CALL "OFFSET",X,Y
```

Change the coordinates of the bottom left hand corner of the screen from 0,0 to X,Y.

```
CALL "GLOAD",A  ) available only
CALL "GSAVE",A  ) for disc systems
```

Load a picture from BASIC cache memory at address A into the HR memory, and vice versa. Picture is in current resolution. In MR, uses page selected by UPDATE. Transfers 15360 (HR) or 7680 (MR) bytes.


```
CALL "SETCOL",I,N
CALL "SETCOL",I,R,G,B
CALL "SETCOL"
```

Similar to COLOUR, except that the effect of this call is delayed until a call to VIEW (q.v.) is made. It enables the setting up of non-standard views and allows a very rapid change of all of the colours.

```
CALL "VIEW"[,P]
```

Transfer the colour changes as specified by SETCOL to the colour lookup table. P (which can be omitted in HR mode) specifies the MR page on which to effect the changes.

```
CALL "UPDATE"[,P],V
```

In MR, there are two pages of memory, which can be written to or displayed independently. P specifies the page which will be written to by subsequent calls to PLOT, LINE, and FILL. It can be omitted in HR mode. In a similar manner, there can be two or four views, or logical pages of memory, which can be written to or displayed independently, if the number of bits/pixel is less than the maximum. This can occur in both HR and MR. V specifies the view to be written to by subsequent calls to PLOT, LINE and FILL.

```
CALL "DISPLAY"[,P],V
```

Specify which page and view is to be displayed. As in UPDATE, P can be omitted in HR mode.

```
CALL "CLEAR"
```

Clear the current page and view, filling it with logical intensity 0.

```
CALL "OFFSET",X,Y
```

Change the coordinates of the bottom left hand corner of the screen from 0,0 to X,Y.

```
CALL "GLOAD",A  ) available only
CALL "GSAVE",A  ) for disc systems
```

Load a picture from BASIC cache memory at address A into the HR memory, and vice versa. Picture is in current resolution. In MR, uses page selected by UPDATE. Transfers 15360 (HR) or 7680 (MR) bytes.

APPENDIX F

COLOUR LOOKUP TABLE

The elements of the colour lookup table that apply in the various modes are tabulated in this Appendix. These tables should be referred to when using SETCOL and VIEW (BASIC) or VIEW (Algol and Fortran). In the three medium resolution modes, all 16 values must be set up by SETCOL before calling VIEW (BASIC), or preset in a user array before calling VIEW (Algol and Fortran). In the two high resolution modes only the first four values need be defined.

HIGH RESOLUTION, 2 BITS/PIXEL (HR2)

The first 4 elements of the colour lookup table correspond directly to intensities 0 to 3.

Table Address	View 0 Intensity
0	0
1	1
2	2
3	3

Values assigned to table addresses 4 to 15 will be ignored.

HIGH RESOLUTION, 1 BIT/PIXEL (HR1)

The first 4 elements of the colour lookup table govern 2 views as follows:

Table Address	View 1 Intensity	View 0 Intensity
0	0	0
1	0	1
2	1	0
3	1	1

Values assigned to table addresses 4 to 15 will be ignored.

MEDIUM RESOLUTION, 4 BITS/PIXEL (MR4)

The 16 elements of the colour lookup table correspond directly to intensities 0 to 15.

MEDIUM RESOLUTION, 2 BITS/PIXEL (MR2)

The 16 elements of the colour lookup table govern 2 views as follows:

Table Address	View 1 Intensity	View 0 Intensity
0	0	0
1	0	1
2	0	2
3	0	3
4	1	0
5	1	1
6	1	2
7	1	3
8	2	0
9	2	1
10	2	2
11	2	3
12	3	0
13	3	1
14	3	2
15	3	3

Thus, to see those pixels that are of intensity 3 and are common to views 0 and 1, set elements 3, 7 and 11 to 15 to be visible, and the rest to background.

MEDIUM RESOLUTION, 1 BIT/PIXEL (MR1)

The 16 elements of the colour lookup table govern 4 views as follows:

Table Address	View 3 Intensity	View 2 Intensity	View 1 Intensity	View 0 Intensity
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Thus, to see those pixels that are common to views 2 and 3, set elements 4 to 15 to be visible, and the rest to background.

INDEX

Note that the detailed descriptions of the graphics calls are arranged alphabetically in Chapter 3. The calls are summarised in Chapter 8.

